SENSOR-BASED ROBOT CONTROL REQUIREMENTS FOR SPACE

DR. RONALD LUMIA
INTELLIGENT CONTROLS GROUP
ROBOT SYSTEMS DIVISION
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
(formerly the National Bureau of Standards)

NASA has begun the development of the Space Station, a permanently manned facility in space, for a variety of scientific goals. One part of this project is the Flight Telerobotic Servicer (FTS) which will help build and maintain the structure. The FTS is envisioned as a two-armed robot with seven degrees of freedom for each arm. When the FTS is launched, it is expected to perform several tasks which include the installation and removal of truss members of the Space Station structure, changeout of a variety of modular units, mating a thermal connector, etc. While the FTS will initially use teleoperation, it is envisioned to become more autonomous as technology advances. In order for the FTS to evolve from teleoperation to autonomy, NASA requires that the NASA/NBS Standard Reference Model (NASREM) be used as the functional architecture for the control system. The quest for autonomy inevitably leads to the need for sophisticated sensors and sensory processing. This paper will explore the requirements for the tasks envisioned for FTS at first launch as well as during its evolution phase and show how those tasks impact research on sensors, sensory processing, and other parts of the FTS control system. Finally, the current state of the NASREM implementation at NIST will be presented.

1. INTRODUCTION

The Flight Telerobotic Servicer (FTS) will be used to build and maintain the Space Station. It is envisioned to be used as a teleoperated device initially. However, it is required to be able to evolve with technology and manifest this evolution by becoming more and more autonomous. While

---

This work is partially supported by funding from NASA under contract S-28187-D and was prepared by U.S. Government employees as part of their official duties and is therefore a work of the U.S. Government and not subject to copyright. Equipment listings in this paper do not imply a recommendation by NIST nor that the equipment is the best for the purpose.

teleoperation is the first step, there has been a deliberate choice not to pursue telepresence.

There is a basic dichotomy in the evolution of FTS activity from the operator's perspective. The FTS can move toward full telepresence or full autonomy. Reaching either extreme, full telepresence or full autonomy, is a long term, possibly unattainable objective. However, it is instructive to examine the possibilities. A system that consists only of telepresence implies that the human remains in the loop for all task steps. In a telepresence system, the human operator performs the tasks, his "presence" being translated to the remote worksite via the technology. This capability is very useful and even necessary for certain applications, especially where the environment is relatively unknown or unstructured. The development of systems which pursue true "telepresence" where the operator is immersed in sensory feedback will require a great deal of R & D.

The nature of the FTS role on Space Station (i.e., assembly, maintenance, inspection, etc.) does not involve tasks of a scientific or totally unstructured nature where the human operator is actually performing some sort of investigation. Therefore, the FTS is choosing to pursue autonomy for several reasons. First, the repetitive chores performed by the robot are less onerous to the operator if the FTS had some autonomous capabilities. Second, the operator does not give up anything since he may break into any level of the NASREM architecture to take control at will. Third, time delays may preclude telepresence for useful applications. For example, in remote satellite servicing, the time delays incurred between the operator and the FTS preclude the use of force reflection and other sophisticated telepresence techniques. An autonomous mode of operation may be the only viable alternative. There may be only a sparse data return, e.g., one image per minute. A telepresence strategy of move and wait would be too slow to be a realistic option.

This paper explores what is required to have a sensor based robot in space. First the tasks originally envisioned for the FTS are presented along with a set of tasks required of an evolved version of the FTS. Then, the problems associated uniquely with space qualification and those associated with the current state of robot technology are presented. This is followed by a description of the NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), which is required by NASA for the control system of the FTS. Finally, the current state of NASREM development at NIST is described.

2. FTS TASKS -- AT FIRST ELEMENT LAUNCH AND ASSEMBLY COMPLETE

The FTS is required to have certain capabilities at First Element Launch (FEL) in support of building and maintaining the Space Station. These tasks are intended to be representative of the range of work which the FTS can perform rather than the limit of its capabilities. The tasks are:

- o Installation and removal of Space Station truss members
- o Installation of a Structural Interface Adapter (SIA) on the truss.
- o Changeout of Space Station Orbital Replaceable Units (ORU).
- o Mating of the Space Station thermal utility connectors.
- o Performance of inspection tasks.
- o Assembly and maintenance of Space Station Electrical Power System Radiator Assembly.

The FTS is required to have, integral to its basic design, the capability of evolving toward autonomous execution of the tasks listed previously. In addition, the FTS should eventually have the capability to perform the following tasks autonomously:

- o Changeout the Hubble Space Telescope (HST) reaction wheel ORU while the HST is secured at the Space

Station or on the Space Shuttle.

- Refuel the Gamma Ray Observatory (GRO) propellant tanks while the GRO is secured at the Space Station or on the STS.
- In-situ servicing and maintenance of Space Station platforms and free fliers.

In order to be able to perform these tasks, a significant amount of sensor and sensor processing technology must be integrated into the FTS system. Sensor capability must be available for:

- Joint control
  For example, the joints of the manipulator could command torques and have sensors which measure actual torque, position, and velocity.

- Vision processing
  Image processing algorithms must be able to cope with extreme variations in lighting.

- Contact force measurement
  During operations such as mating a connector, it is important to be able to measure the forces generated by the robot in contact with the environment. Without this information, there is no way for a control algorithm to correct for errors. This would probably require the use of a force/torque sensor.

- Safety
  Since safety is extremely important, sensors which redundantly measure important parameters must be included in the FTS system. Example of these sensors include robot joint position, proximity of astronauts to the FTS workspace, etc.

The above tasks, as well as the sensors associated with the tasks, will be performed in space. This implies a fairly complex FTS, especially in the requirement for sufficient compute power. The next two sections explore some of the problems in sending the FTS into space.

3. SPACE QUALIFICATION PROBLEMS ASSOCIATED WITH ROBOT SYSTEMS

Space qualification is a rigorous process to ensure that the systems which are sent into the harsh space environment will work reliably. It is well beyond the scope of this paper to explore the technical details of any particular problem. However, this section will hopefully provide the reader with an appreciation of the complexity associated with sending a system into space.

The first problem is associated with materials science. The materials which are used to construct robots and the required ancillary equipment must be able to withstand the harsh space environment and function as expected. The materials must be able to resist thermal fatigue caused by the cycles of extreme heat and cold. Furthermore, they must not outgas, i.e., release gases into space.

The space qualification of electronic components, especially computers, presents a formidable challenge. Most electronic circuitry is sensitive to two types of radiation. The first type is the background radiation which is significantly higher in space than on Earth due to the atmosphere. Consequently, all electronic circuitry must be able to withstand a certain level of total dose radiation. This is tested in the qualification process and a level of radiation hardness must be met. The second type of radiation is caused by cosmic rays, with enough energy to change a bit, passing through the circuit. This phenomenon is known as a single event upset (SEU). Clearly, the circuitry must either be insensitive to the SEU or must detect and react to it. Unfortunately, both the radiation-hard and SEU issues often involve the redesign of the manufacturing processes for the integrated circuits.

For mission critical activities or when human safety is at risk, the electronics are required to be "two fault tolerant." If any fault occurs, the system can still operate. If a second fault develops anywhere in the system, then the system fails in a safe manner. This requirement has major implications for redundancy and protocols of switching between subsystems after a fault is detected. This is clearly an area of continuing research.

Thermal considerations play an important role in the total system design since removing heat by convection is not an option in space. Motors, for example, must be capable of either conducting or radiating the heat generated. It is possible to use passive methods for thermal control but often the active methods using fluid loops are more effective. However, the active loops often require hazardous chemicals such as ammonia compounds and are a great deal more complex to handle. Dangerous chemicals invoke safety rules which may preclude servicing in the pressurized unit and extra-vehicular servcing limits the types of possible repairs.

If robots are going to operate in space without umbilical cords for any period of time, batteries are required. The batteries must operate in a vacuum, be insensitive to temperature variations, store significant energy, weigh as little as possible, and take up a small volume.

Simulation has often been used with great success to determine the value of one approach over another. The problem with simulation, of course, is that the simulation may not represent reality sufficiently, i.e., the model may lack fidelity. On Earth, it is possible to simulate first and then empirically test the system to determine the accuracy of the model. For space, the situation is more complex because it is very difficult to predict exactly how the robot system will behave without gravity.

This list of problems for the space qualification of robot systems is by no means complete. However, the number of significant issues should provide some level of appreciation for the complexity associated with putting robots in space.

4. GENERIC ROBOT TECHNOLOGY PROBLEMS

The second class of problems associated with space robotics is also present in ground based automata. The solutions to this class of problems will advance the state-of-the-art in robotics.

Control methods for robots have traditionally centered on position control where a robot is programmed to follow a predefined path. While this approach has proved quite useful for certain applications in industrial automation, it is not entirely satisfactory for factories and therefore is probably also unsuitable for space utilization. The cost associated with errors in space is enormous and more advanced control methods, such as impedance control [1], appear to hold more promise. However, it is not entirely clear which advanced control algorithm is best for a given application. There is a need to study a set of alternative algorithms in the execution of a given battery of tasks. Knowledge of which algorithms work on which tasks and the reasons why is essential to improve robot control technology.

A second problem area in robot technology is associated with modeling the workspace of the robot. A representation of the robot workspace is required to allow the robot to operate with known objects in a sensible fashion. Many alternatives for this representation exist and must be tested in a systematic way to ascertain which approach is most appropriate. A robot in space operates in a reasonably unstructured environment. Although it can be argued that everything sent into space for the Space Station is man-made and a CAD model is available, there will always be slight discrepancies which will make real objects differ from their models. Since it is possible that such a disparity could result in a catastrophe, such as an object which is larger than its model being pushed through a satellite, a highly calibrated workspace is nearly impossible. The workspace representation can be close, but sensors are required to prevent disasters. Sensory processing, therefore, must be an integral part of a robotic system and interact effectively with the model of the robot workspace so that the algorithms controlling robot motion can be as efficacious as possible.

Sensory processing presents other demanding challenges for robot systems in terms of quality and processing rate. It is well known that lighting is crucial for success in industrial implementations of computer vision. Shadows, specular reflection, lens distortion, etc., will have even more impact in space since the environment cannot be controlled as well as in a factory. The speed of sensory processing is also critical because it can limit the rate the control system can move the robot in response to stimuli. The improvements required for sensory processing in space will increase the available knowledge and ultimately benefit all robot systems.

The last area is concerned with how the operator commands a robot, the operator interface. In teleoperation, there are several issues. The first issue is whether the control device, or master, is kinematically similar to or different from the telerobot. The amount of computation required to control the kinematically similar master is much lower than that required for the kinematically dissimilar master. However, the production of a kinematically similar master is required for each new robot.

Another issue in operator control is kinesthetic feedback. Without force reflection, the operator sends commands to the telerobot but cannot perceive the effect of those commands as the telerobot moves in its workspace. With force reflection, the operator can "feel" the reaction of the telerobot. While it is generally desirable to allow the operator more feedback from the telerobot through force reflection, the control is more difficult and could potentially result in instability.

5. NASA/NBS STANDARD REFERENCE MODEL FOR TELEROBOT CONTROL SYSTEM ARCHITECTURE (NASREM)

The fundamental paradigm of the control system is shown in Figure 1. The control system architecture is a three legged hierarchy of computing modules, serviced by a communications system and a global memory. The task decomposition modules perform real-time planning and task monitoring functions; they decompose task goals both spatially and temporally. The sensory processing modules filter, correlate, detect, and integrate sensory information over both space and time in order to recognize and measure patterns, features, objects, events, and relationships in the external world. The world modeling modules answer queries, make predictions, and compute evaluation functions on the state space defined by the information stored in global memory. Global memory is a database which contains the system's best estimate of the state of the external world. The world modeling modules keep the global memory database current and consistent.

The first leg of the hierarchy consists of task decomposition modules which plan and execute the decomposition of high level goals into low level actions. Task decomposition involves both a temporal decomposition (into sequential actions along the time line) and a spatial decomposition (into concurrent actions by different subsystems). Each task decomposition module at each level of the hierarchy consists of a job assignment manager, a set of planners, and a set of executors. These decompose the input task into both spatially and temporally distinct subtasks.

The second leg of the hierarchy consists of world modeling modules which model (i.e., remember, estimate, predict) and evaluate the state of the world. The "world model" is the system's best estimate and evaluation of the history, current state, and possible future states of the world, including the states of the system being controlled. The "world model" includes both the world modeling modules and a knowledge base stored in a global memory database where state variables, maps, lists of objects and events, and attributes of objects and events are maintained. The world model maintains the global memory knowledge base by accepting information from the sensory system, provides predictions of expected sensory input to the corresponding sensory system modules, based on the state of the task and estimates of the external world, answers "What is?" questions asked by the executors in the corresponding task decomposition modules, and answers "What if?" questions asked by the planners in the corresponding task decomposition modules.

The third leg of the hierarchy consists of sensory processing sensory system modules. These recognize patterns, detect events, and filter and integrate sensory information over space and time. The sensory system modules at each level compare world model predictions with sensory observations and compute correlation and difference functions. These are integrated over time and space so as to fuse sensory information from multiple sources over extended time intervals. Newly detected or recognized events, objects, and relationships are entered by the world modeling modules into the world model global memory database, and objects or relationships perceived to no longer exist are removed. The sensory system modules also contain functions which can compute confidence factors and probabilities of recognized events, and statistical estimates of stochastic state variable values.

The control architecture has an operator interface at each level in the hierarchy. The operator interface provides a means by which human operators, either in the space station or on the ground, can observe and supervise the telerobot. Each level of the task decomposition hierarchy provides an interface where the human operator can assume control. The task commands into any level can be derived either from the higher level task decomposition module, from the operator interface, or from some combination of the two. Using a variety of input devices, a human operator can enter the control hierarchy at any level, at any time of his choosing, to monitor a process, to insert information, to interrupt automatic operation and take control of the task being performed, or to apply human intelligence to sensory processing or world modeling functions.

The sharing of command input between human and autonomous control need not be all or none. It is possible in many cases for the human and the automatic controllers to simultaneously share control of a telerobot system. For example, in an assembly operation, a human might control the position of an end effector while the robot automatically controls its orientation. For a more detailed description of NASREM, see [2].

6. NIST IMPLEMENTATION OF NASREM

In order to implement a functional architecture, especially one like NASREM which allows evolution with technology, the interfaces must be carefully defined. Although the NASREM functional architecture specifies the purpose of each module in the control system hierarchy, it does not completely specify the interfaces between modules. This section will describe the method by which the interfaces for the SERVO level of the hierarchy have been defined. The method involves gathering all of the algorithms available for SERVO level control, dividing each algorithm into the parts which inherently belong to task decomposition, world modeling, and sensory processing, and then deriving the interfaces which will support these algorithms. Any design, however, must constrain the problem sufficiently so that detailed interfaces can be devised.

With this in mind, the Servo Level design was based on a fundamental control approach which computes a motor command as a function of feedback system state y, desired state (attractor) y_d, and control gains. In this approach, the gains are coefficients of a linear combination of state errors $(y-y_d)$. The system state and its attractor are composed from the physical quantities to be controlled, (i.e., position, force, etc.,) and can be expressed in an arbitrary coordinate system. This type of algorithm is the basis for almost all manipulator control schemes [3]. However, this basic algorithm is inadequate for controlling the gross aspects of manipulator motion, as described in [4]. The algorithm can provide "small" motions so that the dynamics of the servo algorithm itself are not significant. This means that the Primitive Level must generate the gross dynamics of the motion through a sequence of inputs to the Servo Level. This can be achieved through an appropriate sequence of either attractor points [3,5] or gain values [4].

Figure 2 depicts the detailed Servo Level design. The task decomposition module at the Servo Level receives input from Primitive in the form of the command specification parameters. The command parameters include a coordinate system specification C_z which indicates the coordinate system in which the current command is to be executed. C_z can specify joint, end-effector, or Cartesian (world) coordinates. Given with respect to this coordinate system are desired position, velocity, and acceleration vectors (z_d, z_d, z_d) for the manipulator, and the desired force and rate of change of force vectors (f_d, f_d). These command vectors form the attractor set for the manipulator. The K's are the gain coefficient matrices for error terms in the control equations. The selection matrices (S,S') apply to certain hybrid force/position control algorithms. Finally, the "Algorithm" specifier selects the control algorithm to be executed by the Servo Level.

When the Servo Level planner receives a new command specification, the planner transmits certain information to world modeling. This information includes an attention function which tells world modeling where to concentrate its efforts, i.e., what information to compute for the executor. The executor simply executes the algorithm indicated in the command specification, using data supplied by world modeling as needed.

The world modeling module at the Servo Level computes model- based quantities for the executor, such as Jacobians, inertia matrices, gravity compensations, Coriolis and centrifugal force compensations, and potential field (obstacle) compensations. In addition, world modeling provides its best guess of the state of the manipulator in terms of positions, velocities, end-effector forces and joint torques. To do this, the module may have to resolve conflicts between sensor data, such as between joint position and Cartesian position sensors.

Sensory processing, as shown in Figure 2, reads sensors relevant to Servo and provides the filtered sensor readings to world modeling. In addition, certain information is transmitted up to the Primitive Level of the sensory processing hierarchy. Primitive uses this information, as well as information from Servo Level world modeling, to monitor execution of its trajectory. Based on this data, Primitive computes the stiffness (gains) of the control, or switches control algorithms altogether. For example, when Primitive detects a contact with a surface, it may switch Servo to a control algorithm that accommodates contact forces.

A more complete description of the Servo Level is available in [3] where the vast majority of the existing algorithms in the literature are described. The same process for developing the interfaces based on the literature has also been performed for the Primitive level and is available in [5]. While the procedure is planned for each level in the hierarchy, the amount of literature support tends to decrease as one moves up the NASREM hierarchy.

Once the interfaces are defined, it is possible to choose a computer architecture and begin to realize the system. While every effort is being made to do the job properly, there is no reason to assume that the implementation at NIST is optimal in any way. It is simply illustrates one realistic method to implement the NASREM architecture.

While a functional architecture is technology independent, its implementation obviously depends entirely on the state-of-the-art of technology. The designer must choose existing computers, buses, languages, etc., and, from these tools, produce a computer architecture capable of performing the <u>functions</u> of the functional architecture. The system must adequately meet the real-time aspects of the controller so that adequate performance is achieved through careful consideration of computer choice, multiple processor real-time operating system, inter-processing communication requirements, tasking within certain processors, etc. For a more detailed description of this methodology, see [6].

The NIST implementation considers two aspects of the software development process: the development environment on which the code is written, debugged, and tested as well as possible, and the target environment where the code for the real-time robot control system is integrated into the system. Figure 3 shows the approach. A network of SUN workstations running UNIX is used for the development environment, sacrificing the speed of the developed code for the ease of development. Once the code is tested as well as possible, it is downloaded to the target system. The target system consists of a VME backplane of several (currently 6) 68020 processors. For rapid iconic image processing, the PIPE system [7] is integrated into the system. The target hardware drives a K-1607 Robotics Research Corp. arm.

From the software side, the multiprocessing operating system used for the target is required to be as simple as possible so that the overhead is minimized. The duties of the operating system are limited to very simple actions such as downloading executable code, starting up the processors, and interprocessor communication. While tasking is not performed at the lower levels of the hierarchy because of the overhead associated with context switches, it is desirable at higher levels in the hierarchy which are not as time critical. NIST researchers are currently investigating three alternatives for tasking: tasking provided by the run-time kernel of the native ADA cross compiler, pSOS tasking, and ADA tasking. Interprocessor communications alternatives including pRISM, sockets, etc., must also be evaluated empirically. The actual application code is written in ADA. Although ADA compilers cannot currently produce code as efficient as other languages such as C, NIST researchers have shown that the gap is steadily decreasing [8].

The application code is developed by programming the processes which achieve the functions associated with the boxes in the functional architecture. The problem then becomes one of assigning each of the processes, such as those shown in Figure 2, to a particular processor. There is a clear trade-off between the cost of the solution and the performance of the system. There are currently no software tools which automatically perform this assignment based on an arbitrary index of performance. The approach at NIST is step-wise refinement of the performance of the system. Given the particular hardware being used, a certain number of processors is chosen arbitrarily. For that configuration, the processes are assigned to the processors. Then, the system is evaluated in terms of its performance. If the performance is unacceptable, the designer has several options. The first option is to add more processors. This alternative is balanced against additional communication required by the processors. Another alternative is to add faster processors or special purpose processors, such as dynamics chips, which optimize particularly compute intensive operations. This trade-off clearly relates to cost. Another alternative is to reassign the processes to the processors in order to balance the workload of each processor. Each of the alternatives can be used by the designer in order to improve the performance of the system. This allows a particular configuration which implements the functional architecture to change with time as improvements in technology are realized.

7. CONCLUSION

The FTS project is the driving force in U.S. space based robots. At first element launch of the Space Station, it will behave as a teleoperated system. However, by using the NASREM architecture, it will be capable of evolving with technology, incorporating greater levels of automation. In order to perform sophisticated autonomous tasks, the FTS must have a significant suite of sensors at first element launch and have the capability to integrate new sensors into its control system as these products become available.

8. REFERENCES

[1] N. Hogan, "Stable Execution of Contact Tasks Using Impedance Control," IEEE International Conference on Robotics and Automation, 1987.

[2] J.S. Albus, R. Lumia, H.G. McCain, "NASA/NBS Standard

Reference Model For Telerobot Control System Architecture (NASREM)," NBS Technote #1235, also as NASA document SS-GSFC-0027.

[3] J. Fiala, "Manipulator Servo Level Task Decomposition," NIST Technote #1255, April 20, 1988.

[4] J. Fiala, "Generation of Smooth Trajectories without Planning," manuscript in preparation.

[5] A.J. Wavering, "Manipulator Primitive Level Task Decomposition," NIST Technote #1256, January 5, 1988.

[6] T. Wheatley, J. Michaloski, and R. Lumia, "Requirements for Implementing Real Time Control Functional Modules on a Hierarchical Parallel Pipelined System," 1989 Conference on Space Telerobotics, Pasadena, January 30, 1989.

[7] E.W. Kent, M.O. Shneier, and R. Lumia, "PIPE," Journal of Parallel and Distributed Computing, Vol. 2, 1985, pp. 50-78.

[8] S. Leake, "A Comparison of Robot Kinematics in ADA and C on Sun and microVAX," Robotics and Automation Session, IASTED, Santa Barbara, CA., May 25-27,1988.

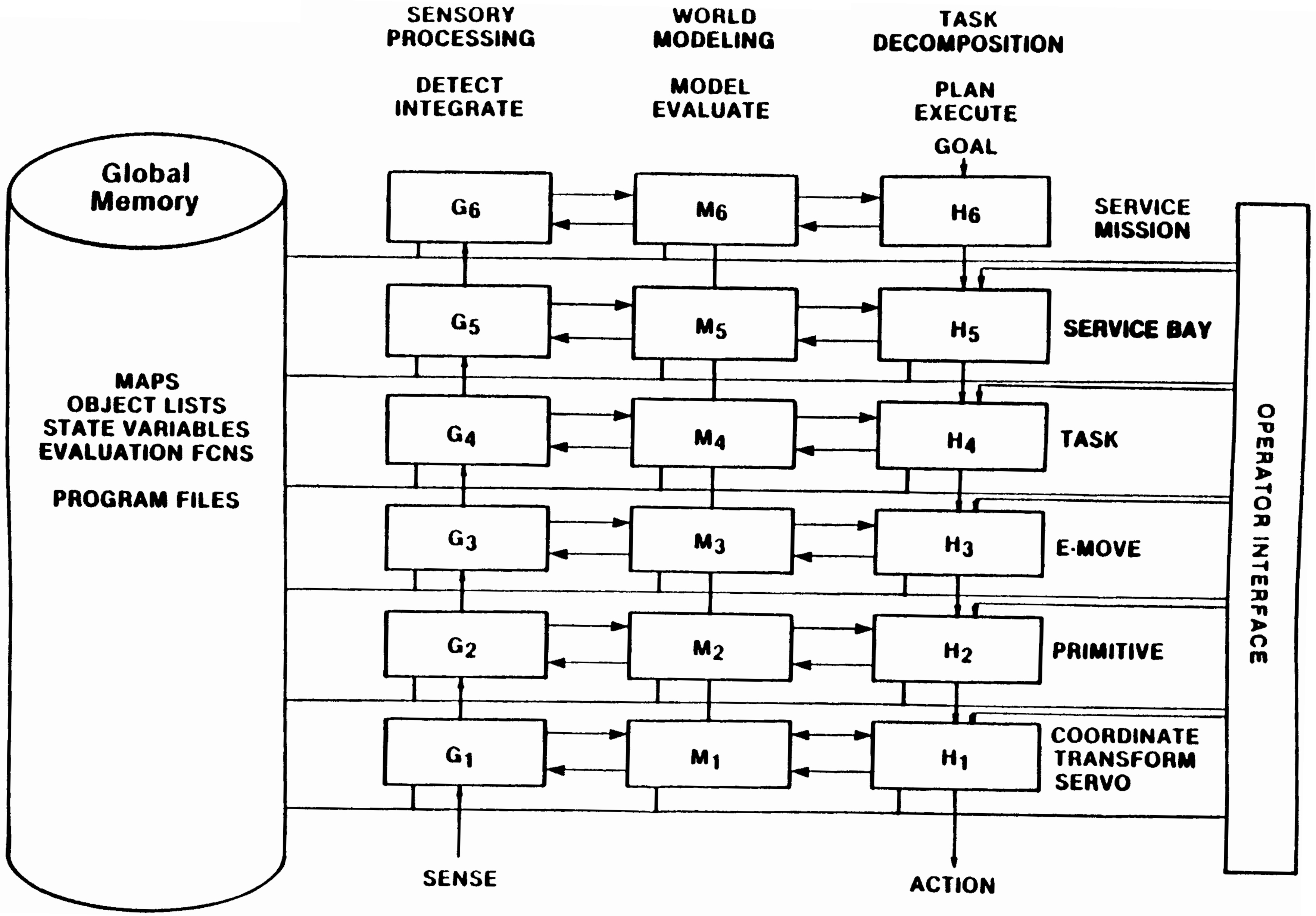

FIGURE 1 - NASA/NBS Standard Reference Model
for Telerobot Control System Architecture (NASREM)

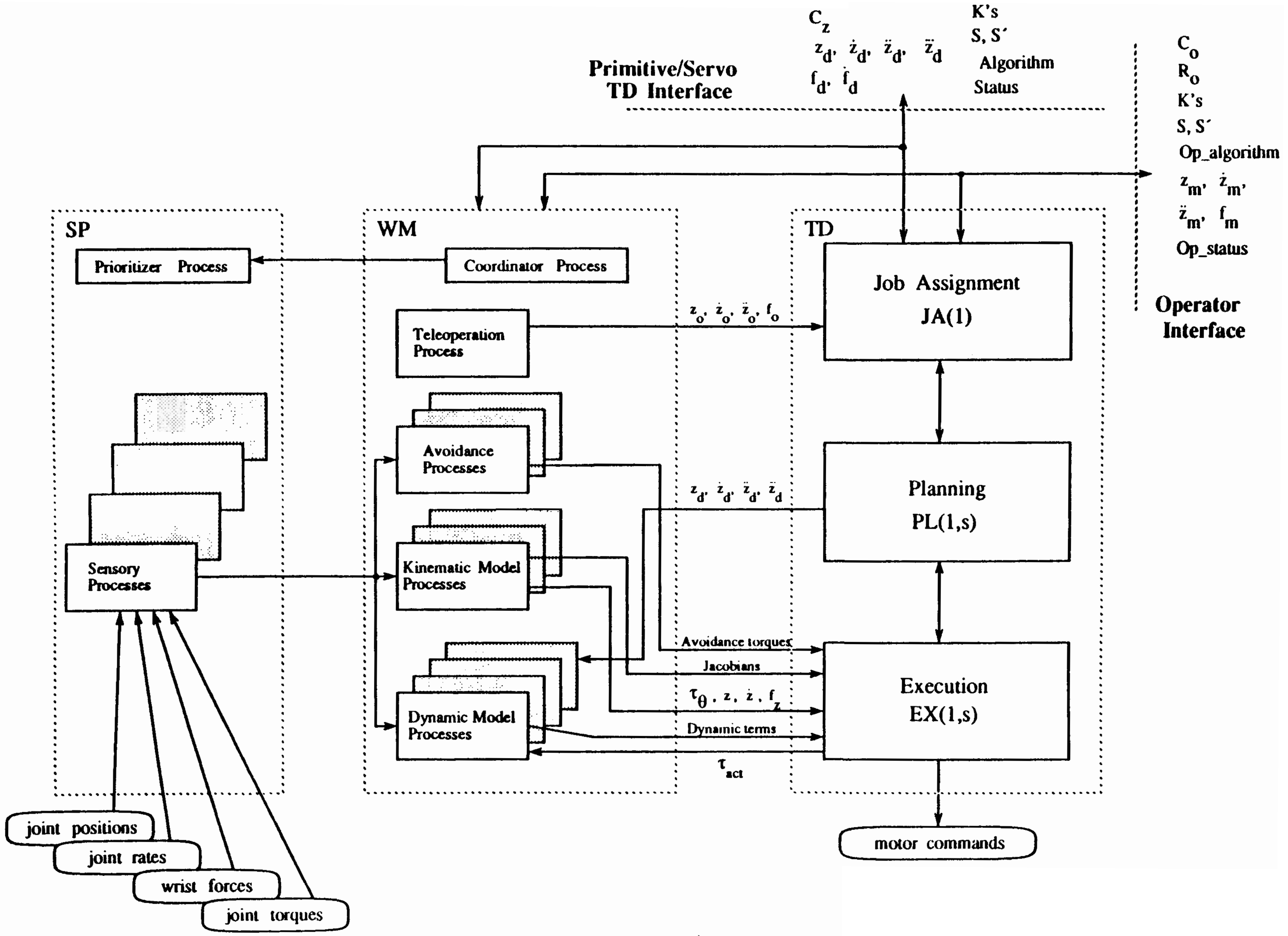

FIGURE 2 - Servo Level Interfaces

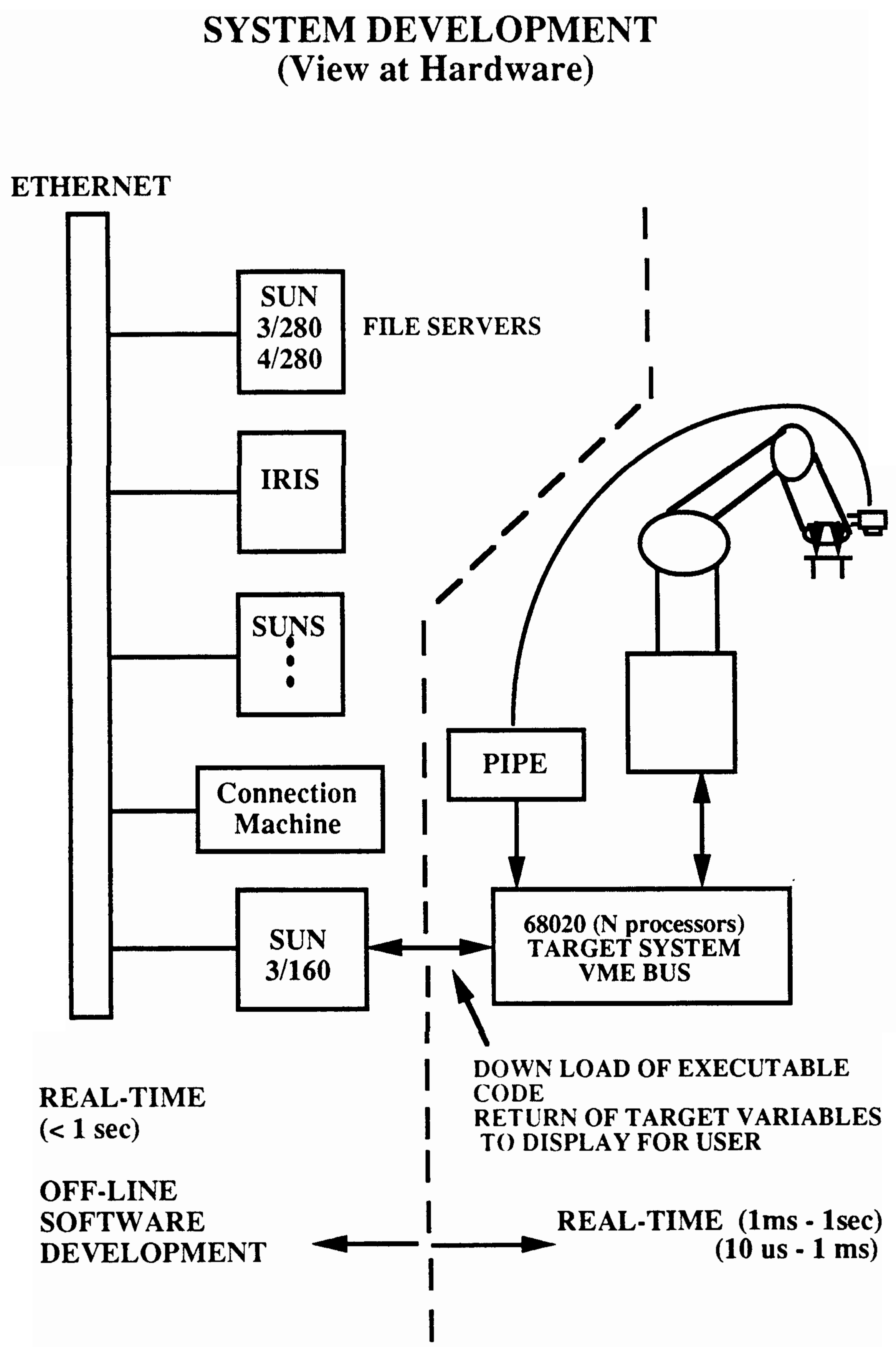

FIGURE 3 - NASREM development at NIST